(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 09081604 A

(43) Date of publication of application: 28 . 03 . 97

(51) Int. Cl G06F 17/50

(21) Application number: 07231579

(22) Date of filing: 08 . 09 . 95

(71) Applicant: TOSHIBA CORP

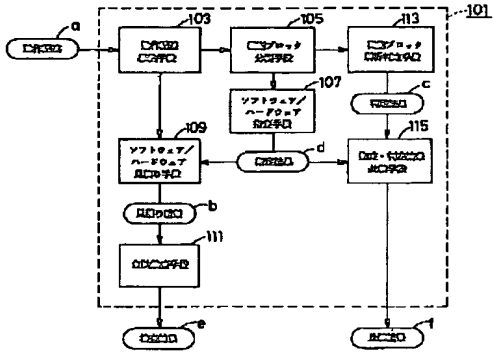
(72) Inventor: AIHARA MASAMI
FUJITA TORU

(54) SOFTWARE/HARDWARE COOPERATION DESIGN SYSTEM AND DESIGN METHOD THEREFOR

(57) Abstract:

PROBLEM TO BE SOLVED: To provide a software/hardware cooperation design system and the design method capable of efficiently finding the division of a software and a hardware for satisfying constraints provided beforehand.

SOLUTION: Respective plural function blocks for which an operation description for describing the operation of a logic system is divided are analyzed and whether or not it is suitable for realization by the hardware is judged by a function block analysis and judgement means 113. The judged result (c) and the specified result of a software/hardware specifying means 107 are compared and a specified/judged result comparison means 115 clearly demonstrates the function block for which specifying different from the judged result (c) is performed.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－81604

(43)公開日　平成９年(1997)３月28日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　17/50			Ｇ０６Ｆ　15/60	６０４Ｈ
				６１２Ｃ
				６３６Ｇ
				６５２Ｒ

審査請求　未請求　請求項の数２　ＯＬ　（全 10 頁）

(21)出願番号　特願平7－231579

(22)出願日　平成７年(1995)９月８日

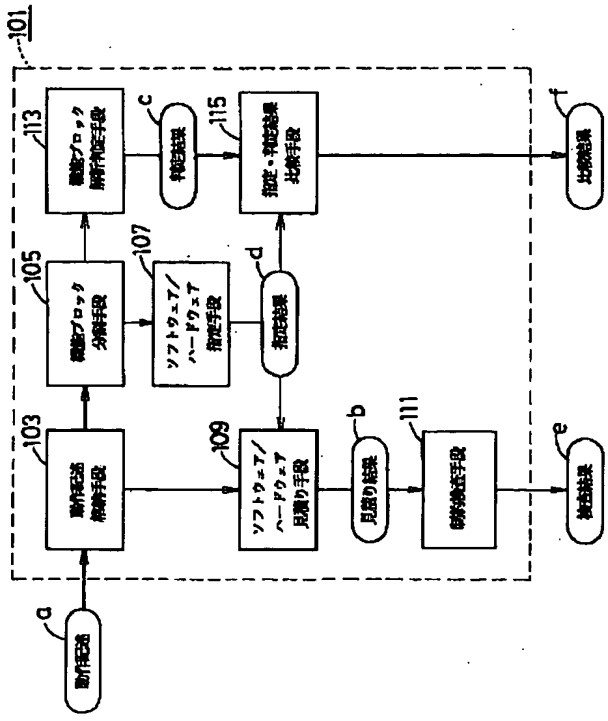
(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地
(72)発明者　相原　雅己
神奈川県川崎市幸区堀川町580番１号　株式会社東芝半導体システム技術センター内
(72)発明者　藤田　透
神奈川県川崎市幸区堀川町580番１号　株式会社東芝半導体システム技術センター内
(74)代理人　弁理士　三好　秀和　（外３名）

(54)【発明の名称】　ソフトウェア／ハードウェア協調設計システム及びその設計方法

(57)【要約】

【課題】　本発明は、上記の事情を考慮してなされたもので、その目的とするところは、予め設けられた制約を満たすようなソフトウェアとハードウェアの分割を効率良く見つけることができるソフトウェア／ハードウェア協調設計システム及びその設計方法を提供することを課題とする。

【解決手段】　論理システムの動作を記述した動作記述を分割した複数の機能ブロックそれぞれを解析し、ハードウェアで実現することに適しているか否かを機能ブロック解析判定手段１１３が判定し、その判定結果ｃとソフトウェア／ハードウェア指定手段１０７の指定結果を比較し、判定結果ｃと異なる指定がされている機能ブロックを指定・判定結果比較手段１１５が明示するように構成されている。



1

【特許請求の範囲】

【請求項１】　ソフトウェアとハードウェアから構成される論理システムを設計するソフトウェア／ハードウェア協調設計システムであって、

ソフトウェアとハードウェアから構成される論理システムの動作を記述した動作記述を格納する動作記述格納手段と、前記動作記述格納手段に格納された前記動作記述を複数の機能ブロックに分割する機能ブロック分割手段と、前記複数の機能ブロックそれぞれに対し、ソフトウェアで実現するかハードウェアで実現するかを指定するソフトウェア／ハードウェア指定手段と、ソフトウェアによる実現を指定された機能ブロックに対してはプログラムサイズと処理時間を見積り、ハードウェアによる実現を指定された機能ブロックに対してはハードウェア量と処理時間を見積るソフトウェア／ハードウェア見積り手段と、前記ソフトウェア／ハードウェア見積り手段の見積り結果が予め設けられた制約を満たすか否かを検査する制約検査手段とを有するソフトウェア／ハードウェア協調設計システムにおいて、

前記複数の機能ブロックそれぞれの動作記述を解析し、ハードウェアで実現することに適しているか否かを判定する機能ブロック解析判定手段と、

前記機能ブロック解析判定手段の判定結果と前記ソフトウェア／ハードウェア指定手段の指定結果を比較し、前記判定結果と異なる指定がされている機能ブロックを明示する指定・判定結果比較手段とを具備することを特徴とするソフトウェア／ハードウェア協調設計システム。

【請求項２】　ソフトウェアとハードウェアから構成される論理システムの動作を記述した動作記述を格納する第１のステップと、

前記第１のステップで格納された動作記述を複数の機能ブロックに分割する第２のステップと、

前記第２のステップで分割された複数の機能ブロックそれぞれに対し、ソフトウェアで実現するかハードウェアで実現するかを指定する第３のステップと、

前記第３のステップでソフトウェアによる実現を指定された機能ブロックに対してはプログラムサイズと処理時間を見積り、ハードウェアによる実現を指定された機能ブロックに対してはハードウェア量と処理時間を見積る第４のステップと、

前記第４のステップの見積り結果が予め設けられた制約を満たすか否かを検査する第５のステップと、

前記第５のステップにおける検査結果において、前記見積り結果が前記制約を満たさない場合には、前記第２のステップで分割された複数の機能ブロックそれぞれの動作記述を解析し、ハードウェアで実現することが適しているか否かを判定する第６のステップと、

前記第６のステップの判定結果と前記第３のステップの指定結果を比較し、前記判定結果と異なる指定がされている機能ブロックを明示する第７のステップと、　前記

2

第７のステップの比較結果に基づき前記第３のステップにおける指定を変更し、前記第５のステップにおいて前記第４のステップの見積り結果が予め設けられた制約を満たすまで、前記第３のステップから前記第７のステップを繰り返す第８のステップとを具備することを特徴とすることを特徴とするソフトウェア／ハードウェア協調設計方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ソフトウェアとハードウェアから構成される論理システムを設計する際、ソフトウェアとハードウェアを同時に設計することを支援するソフトウェア／ハードウェア協調設計システムに関する。

【０００２】

【従来の技術】論理システムの実現を行う場合には、ハードウェアにより実現する方法若しくは汎用プロセッサを利用してソフトウェアにより実現する方法が一般的に用いられている。ところが、近年の微細加工技術の発展に伴いＬＳＩが大規模化してきているために、上記２つの方法を併用する場合が多くなってきている。このような場合、従来では、まず最初に設計者の経験等によりハードウェア部分とソフトウェア部分の切り分けを行ない、その後でハードウェアとソフトウェアをそれぞれ独立に設計するという手順で設計が行われていた。

【０００３】しかし、この手順では、設計が進展してハードウェアの全体像が具体的になった段階で、予め設けられた論理システムの制約とのミスマッチが生じる可能性が高く、仮に、このようなミスマッチが生じた場合には、ハードウェアで実現されている機能ブロックをソフトウェアによる実現に変更してハードウェア量を削減する、ソフトウェアで実現されている機能ブロックをハードウェアによる実現に変更して処理の高速化を図る、というように最初に設計者の経験等により行ったソフトウェアとハードウェアの切り分けを変更しなければならなくなり、多くの場合、最初から設計をやり直す必要が生じてしまうのである。このため、設計期間が長くなってしまうという問題が起きていた。

【０００４】この問題を解決するため、論理システムの動作機能を記述した動作記述を用いて、ソフトウェアに切り分けされた部分に対してはプログラムサイズと処理時間を、ハードウェアに切り分けされた部分に対してはハードウェア量と処理時間を設計を始める前に見積ることにより、その見積り結果が予め設けられた論理システムの制約を満たすようにソフトウェアとハードウェアの切り分けの変更を繰り返し、制約を満たすことが確認できた段階で初めて実際の設計を開始するというソフトウェア／ハードウェア協調設計と呼ばれる設計手順が提案されている。

【０００５】この手順によれば、設計を始める前に、予

め制約を満たすようソフトウェアで実現する機能ブロックとハードウェアで実現する機能ブロックを分けることができるので、上述した設計のやり直しによる設計時間の増大を解決することができる。

【０００６】しかしながら、この手順では、制約を満たすようにソフトウェアとハードウェアの切り分けの変更を行う際に、制約を満たす切り分けが見つかるまで、設計者がいろいろと組合せを変更しなければならないが、例えばハードウェア量を減らしたい場合にどの機能ブロックをハードウェアからソフトウェアに変えたら効果があるのか、あるいは、処理スピードを速くしたい場合にどの機能ブロックをソフトウェアからハードウェアに変えたら効果があるのかということを判断することができないので、設計者は考え得るすべての組合せを逐次的に試していかなければならなず、効率が非常に悪かった。

【０００７】

【発明が解決しようとする課題】以上説明したように、設計者の経験に基づいてソフトウェアとハードウェアを切り分けし、その後、ソフトウェアとハードウェアを別々に設計するという、従来の設計手順では、設計がある程度進展した段階になって制約とのミスマッチが生じる可能性が高く、結果として多くの場合は最初から設計をやり直す必要が生じるので、設計期間が長くなるという問題があった。

【０００８】一方、ソフトウェア部分及びハードウェア部分それぞれのプログラムサイズ、ハードウェア量、処理時間を見積り、その見積り結果が制約を満たすようなソフトウェアとハードウェアの切り分けを見つけてから、ソフトウェアとハードウェアそれぞれの設計を開始するというソフトウェア／ハードウェア協調設計手順では、上記従来の手順に比べて設計がある程度進んだ段階で設計のやり直しをすることがなくなるので設計期間を短縮する効果はあるが、ソフトウェアとハードウェアとの切り分けを決める際に、考え得る組合せすべてを逐次的に試さなければならないため切り分けの決定に時間がかかり、設計期間の点で効率が悪いという問題が生じていた。

【０００９】そこで、本発明は、上記の事情を考慮してなされたもので、その目的とするところは、予め設けられた制約を満たすようなソフトウェアとハードウェアの分割を効率良く見つけることができるソフトウェア／ハードウェア協調設計システム及びその設計方法を提供することにある。

【００１０】

【課題を解決するための手段】上記目的を達成するため本発明は、ソフトウェアとハードウェアから構成される論理システムを設計するソフトウェア／ハードウェア協調設計システムであって、ソフトウェアとハードウェアから構成される論理システムの動作を記述した動作記述を格納する動作記述格納手段と、前記動作記述格納手段に格納された前記動作記述を複数の機能ブロックに分割する機能ブロック分割手段と、前記複数の機能ブロックそれぞれに対し、ソフトウェアで実現するかハードウェアで実現するかを指定するソフトウェア／ハードウェア指定手段と、ソフトウェアによる実現を指定された機能ブロックに対してはプログラムサイズと処理時間を見積り、ハードウェアによる実現を指定された機能ブロックに対してはハードウェア量と処理時間を見積るソフトウェア／ハードウェア見積り手段と、前記ソフトウェア／ハードウェア見積り手段の見積り結果が予め設けられた制約を満たすか否かを検査する制約検査手段とを有するソフトウェア／ハードウェア協調設計システムにおいて、前記複数の機能ブロックそれぞれの動作記述を解析し、ハードウェアで実現することに適しているか否かを判定する機能ブロック解析判定手段と、前記機能ブロック解析判定手段の判定結果と前記ソフトウェア／ハードウェア指定手段の指定結果を比較し、前記判定結果と異なる指定がされている機能ブロックを明示する指定・判定結果比較手段とを具備することを特徴とする。

【００１１】なお、前記機能ブロック解析判定手段は、各機能ブロックの動作記述が、パイプライン処理を行うハードウェアに適しているか否か、及び、並列処理を行うハードウェアに適しているか否かにより判定することができる。

【００１２】

【発明の実施の形態】以下、図面を用いて本発明の実施の形態を説明する。

【００１３】図１は、本発明の実施形態に係るソフトウェア／ハードウェア協調設計システムの一構成例を示す図である。

【００１４】図１に示すソフトウェア／ハードウェア協調設計システム１０１は、ソフトウェアとハードウェアから構成される論理システムの動作を記述した動作記述ａを格納する動作記述格納手段１０３と、動作記述格納手段１０３に格納された動作記述ａを複数の機能ブロックに分割する機能ブロック分割手段１０５と、前記複数の機能ブロックそれぞれに対し、ソフトウェアで実現するかハードウェアで実現するかを指定するソフトウェア／ハードウェア指定手段１０７と、ソフトウェアによる実現を指定された機能ブロックに対してはプログラムサイズと処理時間を見積り、ハードウェアによる実現を指定された機能ブロックに対してはハードウェア量と処理時間を見積るソフトウェア／ハードウェア見積り手段１０９と、ソフトウェア／ハードウェア見積り手段１０９の見積り結果ｂが予め設けられた制約を満たすか否かを検査し、検査結果ｅを出力する制約検査手段１１１と、前記複数の機能ブロックそれぞれの動作記述ａを解析し、ハードウェアで実現することに適しているか否かを判定する機能ブロック解析判定手段１１３と、機能ブロック解析判定手段１１３の判定結果ｃとソフトウェア／

ハードウェア指定手段１０７の指定結果ｄを比較し、判定結果ｃと異なる指定がされている機能ブロックを明示することで比較結果ｆを出力する指定・判定結果比較手段１１５から構成されている。

【００１５】次に、動作記述ａの具体例を用いて本発明の実施形態に係るソフトウェア／ハードウェア協調設計システムの動作を図面を参照しながら説明する。なお、図２は、図１に示す動作記述ａの一例を示す図であり、機能ブロック２０１、２０２、２０３から構成されている。また、図３は、本発明の実施形態に係るソフトウェア／ハードウェア協調設計の手順を示すフローチャートである。

【００１６】図３において、まず、動作記述格納手段１０３は、動作記述ａを格納する（ステップ１）。

【００１７】次に、機能ブロック分割手段１０５は、ステップ１で動作記述格納手段１０３に格納された動作記述ａを機能ブロック２０１、２０２、２０３に分割する（ステップ２）。

【００１８】次に、ソフトウェア／ハードウェア指定手段１０７は、ステップ２で動作記述ａを分割した機能ブロック２０１、２０２、２０３それぞれに対して、ソフトウェアで実現するかハードウェアで実現するかを指定する。図４は、機能ブロック２０１、２０２、２０３それぞれの実現方法についての指定結果の一例を示す図である。ここでは、機能ブロック２０１と機能ブロック２０２にはソフトウェアによる実現、機能ブロック２０３にはハードウェアによる実現をそれぞれ指定している（ステップ３）。

【００１９】次に、ソフトウェア／ハードウェア見積り手段１０９は、ステップ３でソフトウェアによる実現を指定された機能ブロック２０１及び２０２に対してはプログラムサイズと処理時間の見積りを行い、一方、ハードウェアによる実現を指定された機能ブロック２０３に対してはハードウェア量と処理時間の見積りを行う（ステップ４）。

【００２０】次に、制約検査手段１１１は、ステップ４でソフトウェア／ハードウェア見積り手段１０９が見積った見積り結果が予め設けられた制約を満たすか否かを検査し、見積り結果が制約を満たす場合には、ステップ３での指定結果に基づきソフトウェアとハードウェアをそれぞれ作成し、所望の論理システムの設計はここで終了する。（ステップ５）。

【００２１】次に、ステップ５において見積り結果が制約を満たさない場合、機能ブロック解析判定手段１１３は、ステップ２で分割された機能ブロック２０１、２０２、２０３それぞれについて、その動作記述を解析し、ハードウェアに適した記述か否かを判定する。図５は、機能ブロック２０１、２０２、２０３それぞれについて、ハードウェアによる実現が適しているか否かの判定結果の一例を示す図である（ステップ６）。なお、この判定方法については後で説明する。

【００２２】次に、指定・判定結果比較手段１１５は、ステップ３でソフトウェア／ハードウェア指定手段１０７が行った指定結果（図４）とステップ６で機能ブロック解析判定手段１１３が行った判定結果（図５）を比較し、判定結果と異なる指定がされている機能ブロックを明示する。図６は、図４に示す指定結果と図５に示す判定結果の比較結果の一例を示す図である（ステップ７）。

【００２３】最後に、図６に示す比較結果に基いて、ステップ２における指定を変更し、ステップ５においてステップ４における見積り結果が制約を満たすまでステップ３からステップ７を繰り返す（ステップ８）。なお、この変更方法についても後で説明する。

【００２４】次に、ステップ６の判定方法について説明する。

【００２５】図２に示す機能ブロック２０１については、まず、変数の参照関係を解析する。値を参照している変数は配列ｄａｔａ、ａａ、ｘ、ｙであり、値を設定している変数は配列ｉｌｌｕ、ａａ、ｘ、ｙであることがわかる。この内、ｘ、ｙは配列のインデックスとしてのみ使用され、ａａは設定／参照の順序関係から一時的に値を保持するために使用されていることもわかる。これにより配列ｄａｔａを機能ブロック２０１の入力、配列ｉｌｌｕを出力とみなす。

【００２６】次に、入力と出力の依存関係からハードウェアに適した記述か否かを判定する。１回のループでｄａｔａ［ｘ］［ｙ］，ｄａｔａ［ｘ］［ｙ－１］，ｄａｔａ［ｘ－１］［ｙ］の３つの値を使って計算しており、また規則的であることから、パイプライン処理を行なうハードウェアに適した記述であると判定する。また、入力のみから出力を計算しておりループ１回毎の計算はそれぞれ独立していることから、並列処理を行なうハードウェアに適した記述であると判定する。

【００２７】上述したように、機能ブロック２０１は並列処理またはパイプライン処理を行うハードウェアに適したものであり、この時、図５に示すように、判定結果は“適している”と表示される。

【００２８】図２に示す機能ブロック２０２については、変数の参照関係を解析した結果、配列ｉｌｌｕを機能ブロックの入力、配列ｖｉを出力、変数ｄｉｃｔはループ内部で参照が先にあり、設定が後にあることから前のループの計算結果を次のループの計算で使うために一時的に保持している変数とみなす。１回のループでｉｌｌｕ［ｍ］［ｎ］の値を使って計算しているが、この場合、ｍ、ｎの値の決め方に規則性がないためパイプライン処理を行なうハードウェアに適した記述ではないと判定する。また、変数ｄｉｃｔにより１回前のループでの計算結果を使用するため、１回毎の計算に依存関係が生じるため並列処理を行なうハードウェアに適した記述で

はないと判定する。従って、機能ブロック２０２はハードウェアに適した記述ではないと判定する。

【００２９】上述したように、機能ブロック２０２は並列処理及びパイプライン処理を行うハードウェアに適したものではないので、図５に示すように、判定結果は“適してない”と表示される。

【００３０】図２に示す機能ブロック２０３についても同様に、変数の参照関係を解析した結果、配列ｖｉを機能ブロック２０３の入力、配列ｖｏを出力、変数ｘｄ、ｙｄもループ内部で参照が先にあり設定が後にあるため前のループの計算結果を次のループでの計算に使うために一時的に保持している変数とみなす。１回のループでｖｉ［ｘ］、ｖｉ［ｘ－１］の値を使って計算しており規則てきであることからパイプライン処理を行なうハードウェアに適した記述であると判定する。しかし、変数ｘｄ、ｙｄにより１回前のループでの計算結果を使用するため、１回毎の計算に依存関係が生じるため並列処理を行なうハードウェアに適した記述ではないと判定する。

【００３１】上述したように、機能ブロック２０３はパイプライン処理を行うハードウェアに適したものであり、図５に示すように判定結果は“適している”と表示される。

【００３２】次に、ステップ８の指定変更方法について説明する。

【００３３】図６において、指定・判定結果比較手段１１５は、ステップ６でハードウェアによる実現が適していると判定されているにもかかわらず、ステップ２でソフトウェアによる実現が指定されている機能ブロック２０１をハイライト表示により明示している。これにより、設計者は、制約を満たすようにソフトウェアで実現する機能ブロックとハードウェアで実現する機能ブロックの指定の変更を行う。

【００３４】例えば、図４に示すソフトウェア／ハードウェア指定では処理時間が長すぎるという理由から制約を満たさない場合、ソフトウェアによる実現を指定されている機能ブロック２０１及び２０３のうち、ハードウェアによる実現が適している機能ブロックをソフトウェアによる実現からハードウェアによる実現に変更すれば、処理時間を短縮することができる。従来では、設計者がこの２つの機能ブロックについて指定をどう変えれば制約が満たされるかを試行錯誤的に試さなければならなかったが、本発明では図６に示す表示を見れば、制約を満たすソフトウェア／ハードウェアの指定を効率良く見つけることが可能である。

【００３５】従って、ハイライト表示されている機能ブロック２０１の指定を変えることで処理時間を短縮することができるのである。なお、この場合、ハードウェア量の制約を考慮して並列処理あるいはパイプライン処理を選択すれば良い。

【００３６】図７は、図２に示す機能ブロック２０１、２０２、２０３それぞれの実現方法についての指定結果の他の一例を示す図である。ここでは、機能ブロック２０１、２０２及び２０３はハードウェアによる実現をそれぞれ指定している。

【００３７】図８は、図７に示す指定結果と図５に示す判定結果を指定・判定結果比較手段１１５により比較し、判定結果と異なる指定をされている機能ブロックを明示した一例を示す図であり、機能ブロック２０２がハードウェアに適した記述でないにもかかわらずハードウェアによる実現が指定されているためハイライト表示されている。

【００３８】例えば、図７に示すソフトウェア／ハードウェア指定ではハードウェア量が多すぎて制約を満たさないとする。この場合、ハードウェアによる実現を指定されている機能ブロックをソフトウェアによる実現に変えることで、ハードウェア量を減らすことができるので、図７に示す表示を見て、ハイライト表示されている機能ブロック２０２をハードウェアによる実現からソフトウェアによる実現に変えることで、処理時間にあまり影響を与えないでハードウェア量を減らすことができる。

【００３９】ここで、図２に示す機能ブロックのハードウェアによる実現の一例を示しておく。

【００４０】図９及び図１０は図２に示す機能ブロック２０１をハードウェアで実現した場合の論理回路の一例を示す図であり、図９は並列処理を行なうハードウェア、図１０はパイプライン処理を行なうハードウェアを示している。また、図１１は図２に示す機能ブロック２０２をハードウェアで実現した場合の論理回路の一例を示す図、図１２は図２に示す機能ブロック２０３をハードウェアで実現した場合の論理回路の一例を示す図である。

【００４１】図９～図１２から明らかなように、図１０及び図１２のパイプライン処理する論理回路のハードウェア量は少ないが、図９の並列処理する論理回路及び図１１のハードウェアに適した記述でない機能ブロックの論理回路はハードウェア量が多いことがわかる。

【００４２】一方、ソフトウェアで実現した機能ブロックは、プロセッサとメモリが必要であるが、他のブロックと共用可能であるために無視するとハードウェア量はゼロとみなせる。また、一般的に、ソフトウェアよりもハードウェアが、逐次処理よりもパイプライン処理のほうが、パイプライン処理よりも並列処理のほうが、処理時間が短くなる傾向がある。

【００４３】従って、機能ブロック２０１及び機能ブロック２０３の動作はパイプライン処理あるいは並列処理を行うハードウェアで実現することにより処理時間を大幅に短縮できる。一方、機能ブロック２０２の動作は逐次的であり、ハードウェアで実現しても処理時間を大幅

に短縮することはできず、また、ハードウェア量も多いので、ソフトウェアによる実現が適しているのである。

【００４４】

【発明の効果】以上詳述したように本発明によれば、論理システムの動作を記述した動作記述を解析して機能ブロックそれぞれに対してハードウェアに適した記述か否かを判定し、設計者が指定した結果と比較することにより、ハードウェアに適した記述にも拘らずソフトウェアによる実現を指定されている機能ブロックあるいはハードウェアに適していない記述にも拘らずハードウェアによる実現を指定されている機能ブロックを明示することができる。これにより、制約が満たされず指定を変更する場合に、どの機能ブロックの指定を変えたらハードウェア量を効果的に削減することができるかあるいは処理時間を効果的に短縮できるかが容易に判定できるため、制約を満たすようなソフトウェアとハードウェアの指定を無駄な試行錯誤をすることなく求めることが可能になり、結果的に設計期間が短縮されるという効果がある。

【図面の簡単な説明】

【図１】本発明の実施形態に係るソフトウェア／ハードウェア協調設計システムの一構成を示す図である。

【図２】図１に示す動作記述ａの一例を示す図である。

【図３】本発明の実施形態に係るソフトウェア／ハードウェア協調設計の手順を示すフローチャートである。

【図４】図２に示す機能ブロック２０１、２０２、２０３それぞれの実現方法についての指定結果の一例を示す図である。

【図５】図２に示す機能ブロック２０１、２０２、２０３それぞれについて、ハードウェアによる実現が適しているか否かの判定結果の一例を示す図である。

＊【図６】図４に示す指定結果と図５に示す判定結果の比較結果の一例を示す図である。

【図７】図２に示す機能ブロック２０１、２０２、２０３それぞれの実現方法についての指定結果の他の一例を示す図である。

【図８】図７に示す指定結果と図５に示す判定結果を指定・判定結果比較手段１１５により比較し、判定結果と異なる指定をされている機能ブロックを明示した一例を示す図である

【図９】図２に示す機能ブロック２０１を並列処理を行うハードウェアで実現した場合の論理回路の一例を示す図である。

【図１０】図２に示す機能ブロック２０１をパイプライン処理を行うハードウェアで実現した場合の論理回路の一例を示す図である。

【図１１】図２に示す機能ブロック２０２をハードウェアで実現した場合の論理回路の一例を示す図である。

【図１２】図２に示す機能ブロック２０３をハードウェアで実現した場合の論理回路の一例を示す図である。

【符号の説明】

１０１　ソフトウェア／ハードウェア協調設計システム
１０３　動作記述格納手段
１０５　機能ブロック分割手段
１０７　ソフトウェア／ハードウェア指定手段
１０９　ソフトウェア／ハードウェア見積り手段
１１１　制約検査手段
１１３　機能ブロック解析判定手段
１１５　指定・判定結果比較手段
２０１、２０２、２０３　機能ブロック

【図４】

機能ブロック	実現方法
２０１	ソフトウェア
２０２	ソフトウェア
２０３	ハードウェア

【図５】

機能ブロック	並列処理	パイプライン	判定結果
２０１	○	○	適している
２０２	×	×	適してない
２０３	×	○	適している

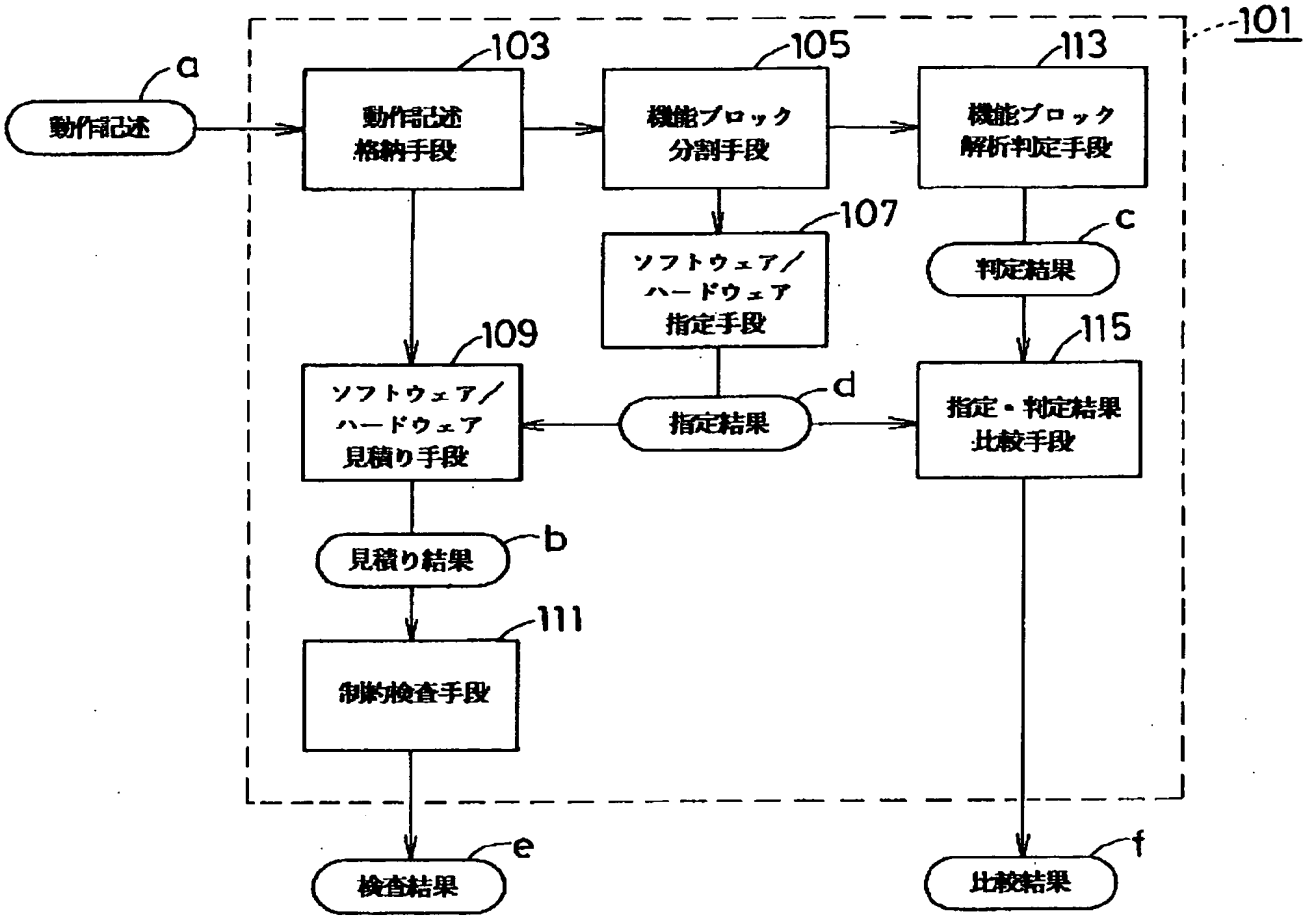
【図７】

機能ブロック	実現方法
２０１	ハードウェア
２０２	ハードウェア
２０３	ハードウェア

【図８】

機能ブロック	実現方法	並列処理	パイプライン	判定結果
２０１	ハードウェア	○	○	適している
２０２	ハードウェア	×	×	適してない
２０３	ハードウェア	×	○	適している

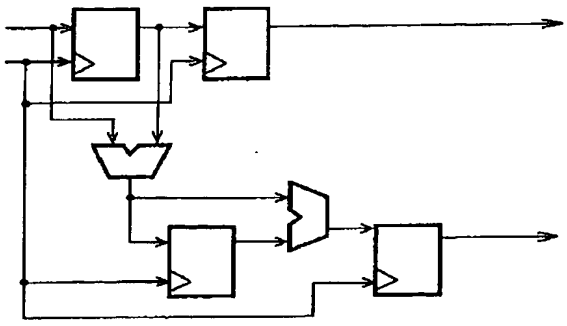
【図１】



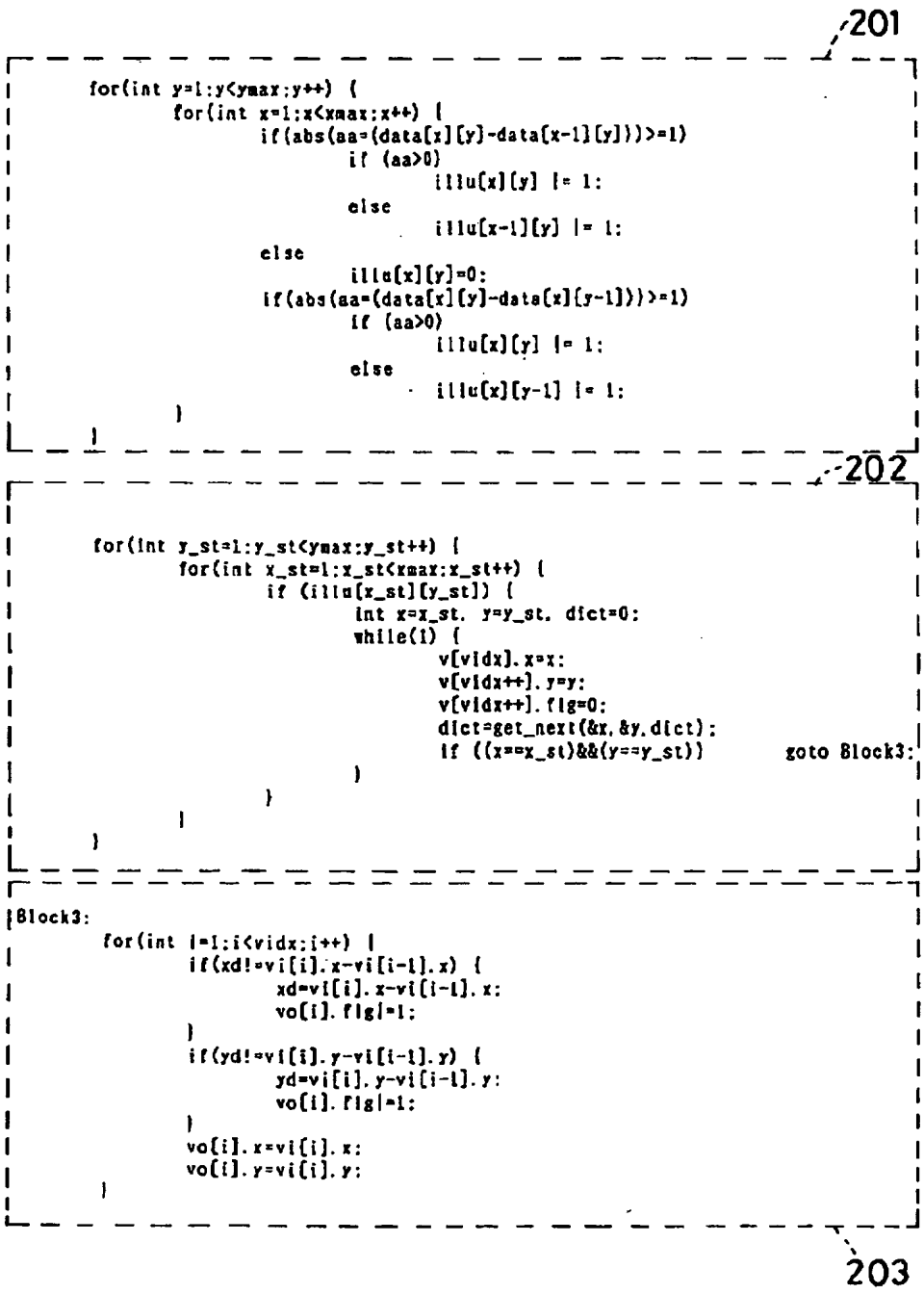
【図６】

機能ブロック	実現方法	並列処理	パイプライン	判定結果
201	ソフトウェア	○	○	通している
202	ソフトウェア	×	×	通してない
203	ハードウェア	×	○	通している

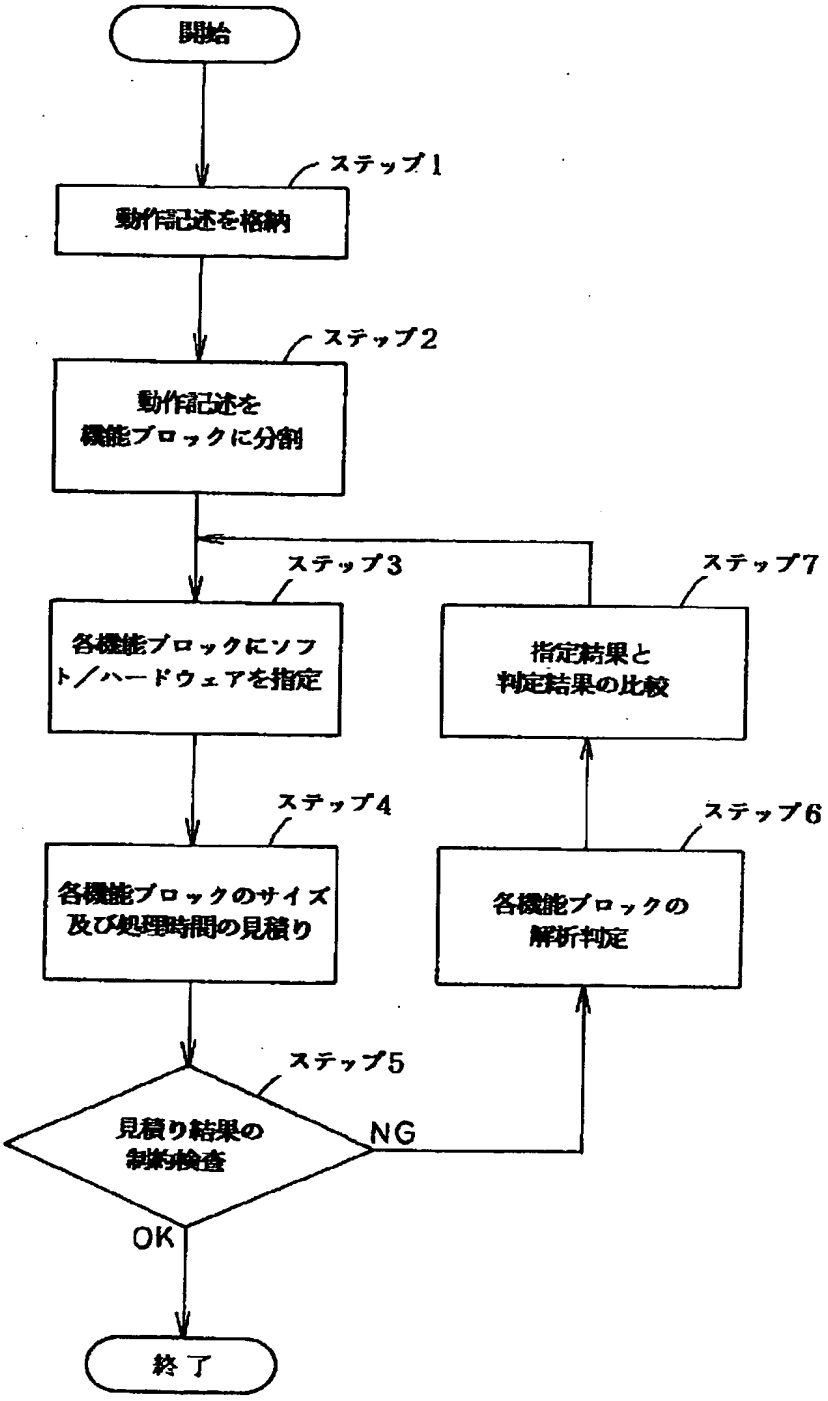
【図１２】



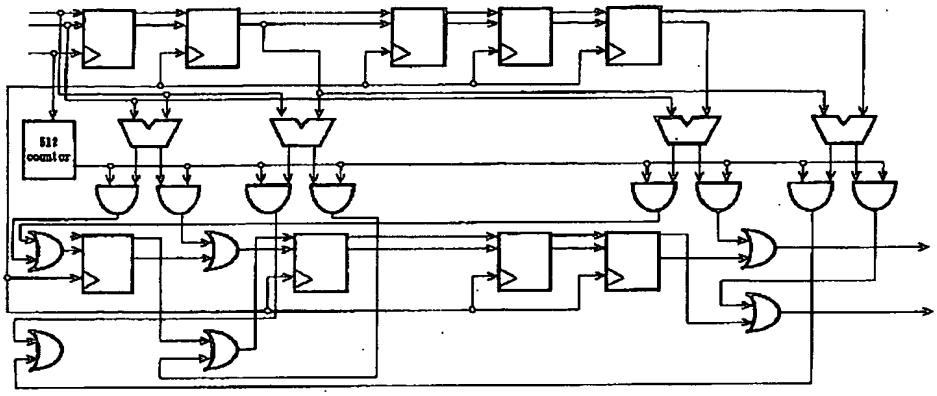
【図２】


201
for(int y=1;y<ymax;y++) {
for(int x=1;x<xmax;x++) {
if(abs(aa=(data[x][y]-data[x-1][y]))>=1)
if (aa>0)
illu[x][y] |= 1;
else
illu[x-1][y] |= 1;
else
illu[x][y]=0;
if(abs(aa=(data[x][y]-data[x][y-1]))>=1)
if (aa>0)
illu[x][y] |= 1;
else
illu[x][y-1] |= 1;
}
}
202
for(int y_st=1;y_st<ymax;y_st++) {
for(int x_st=1;x_st<xmax;x_st++) {
if (illu[x_st][y_st]) {
int x=x_st, y=y_st, dict=0;
while(1) {
v[vidx].x=x;
v[vidx++].y=y;
v[vidx++].flg=0;
dict=get_next(&x,&y,dict);
if ((x==x_st)&&(y==y_st)) goto Block3;
}
}
}
}
Block3:
for(int i=1;i<vidx;i++) {
if(xd!=vi[i].x-vi[i-1].x) {
xd=vi[i].x-vi[i-1].x;
vo[i].flg|=1;
}
if(yd!=vi[i].y-vi[i-1].y) {
yd=vi[i].y-vi[i-1].y;
vo[i].flg|=1;
}
vo[i].x=vi[i].x;
vo[i].y=vi[i].y;
}
203

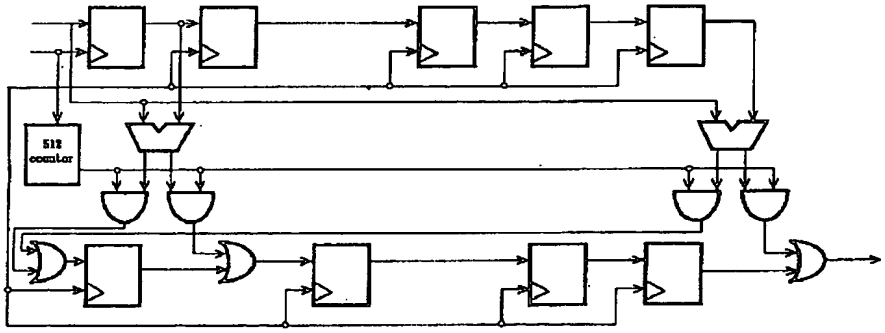
【図３】


開始
ステップ1
動作記述を格納
ステップ2
動作記述を
機能ブロックに分割
ステップ3
各機能ブロックにソフト／ハードウェアを指定
ステップ4
各機能ブロックのサイズ及び処理時間の見積り
ステップ5
見積り結果の制約検査
NG
OK
ステップ6
各機能ブロックの解析判定
ステップ7
指定結果と判定結果の比較
終了

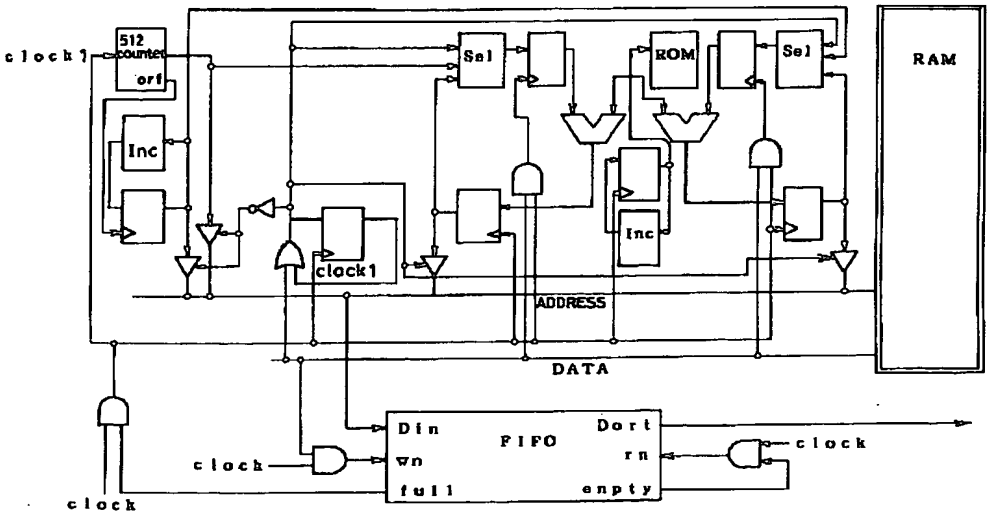
【図９】


512
counter

【図１０】


512
counter

【図１１】


clock1
512
counter
ovf
Inc
Sel
ROM
Sel
RAM
Inc
clock1
ADDRESS
DATA
Din
FIFO
Dout
wn
rn
full
enpty
clock
clock
clock